OpenGL: Integrated Graphics 2

Session 506

OpenGL: Integrated Graphics 2

Ken Dyke
Sr. Mad Scientist

Topics to Be Covered

- Integrating 2D content with OpenGL
- Compositing with OpenGL
- Image processing effects

Integrating 2D Content

- Quartz
- QuickTime

Quartz

- Create ARGB CGBitmapContext

```
cctx = CGBitmapContextCreate(colorTextureData,
  colorTextureWidth, colorTextureHeight, 8,
  colorTextureBytesPerRow,
  CGColorSpaceCreateDeviceRGB(),
  kCGImageAlphaPremultipliedFirst);
```

- Use `GL_BGRA` and `GL_UNSIGNED_INT_8_8_8_8_REV`
- CG Content is always premultiplied

Demo

Quartz 2D Integration

Kenneth Dyke
Sr. Mad Scientist

QuickTime

- Performance is critical
  - SD is 20MB/sec
  - HD nearly 120MB/sec
- We need to use fast OpenGL data path
- No CPU copies

APPLE_client_storage

- Lets OpenGL use application memory for texture source data
- Eliminates CPU copy and saves memory
- Example

```
glPixelStorei(GL_UNPACK_CLIENT_STORAGE_APPLE,
                GL_TRUE);
glBindTexture(GL_TEXTURE_2D, textureName);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA,
   width, height, 0, GL_RGBA, GL_UNSIGNED_BYTE,
   textureData);
```

APPLE_texture_range

- Provides hint to driver for how to store texture data
- New **TEXTURE_STORAGE_HINT_APPLE** enum for glTexParameteri();

STORAGE_PRIVATE_APPLE

- Driver keeps independent copy from OpenGL framework
    - This is OpenGL's default mode
- Allows for fully asynchronous changes to texture data

STORAGE_SHARED_APPLE

- Driver tells hardware to access OpenGL framework's copy directly
- This effectively forces AGP or PCI texturing
- Lowers VRAM usage at the possible expense of texturing performance
- OpenGL framework must do internal synchronization for texture updates

STORAGE_CACHED_APPLE

- Driver caches OpenGL framework's copy in video memory
- Uses high speed DMA texture upload
- Consumes more VRAM, but provides maximum texturing performance
- Good if you will texture from data more than once
- Also requires internal synchronization

Interactions

- Using either extension alone doesn’t require external synchronization
- Using both together is where it gets tricky
  - Application must synchronize with GL before changing data

How to Synchronize

- glFinish()
  - Waits for the entire pipeline to complete; Not optimal
- glSetFence()/glFinishFence()
  - Waits for specific point in command stream, but has performance cost
- glFinishObject()
  - Waits for access to a specific object to be done
  - glFinishObject(GL_TEXTURE, textureName);

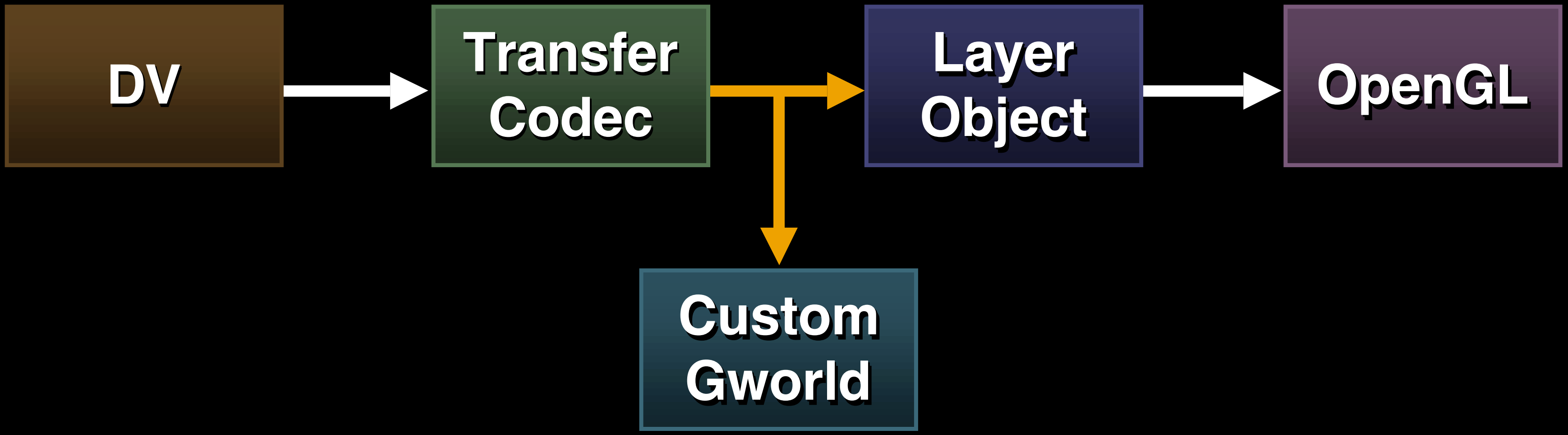
QuickTime Synchronization—The Problem

- High-level QT callbacks are too late
- We need to know before and after QuickTime modifies the data

QuickTime Synchronization— The Solution

- Custom Transfer codec
    - We get lock/unlock calls
    - We can choose which formats to support
- Custom pixel format (OGLX)
    - Forces ICM to use our transfer codec
    - Pixmap base pointer as void *

Block Diagram of the Solution



OpenGL YUV Transfer Codec Details

- Advertises ‘2vuy’ and ‘yuvs’ formats
- Mostly Straightforward implementation
    - Main functionality is to hook into LockBits/UnlockBits
    - Treats GWorld base pointer as Objective-C id

What About RGB Data?

- ICM tries to handle ‘raw’ format directly
    - Will bypass our transfer codec
- Workaround is to use another custom transfer codec
    - ‘raw’ to ‘OGLR’—tricks ICM into letting us do the work

Demo

OpenGL Transfer Codecs

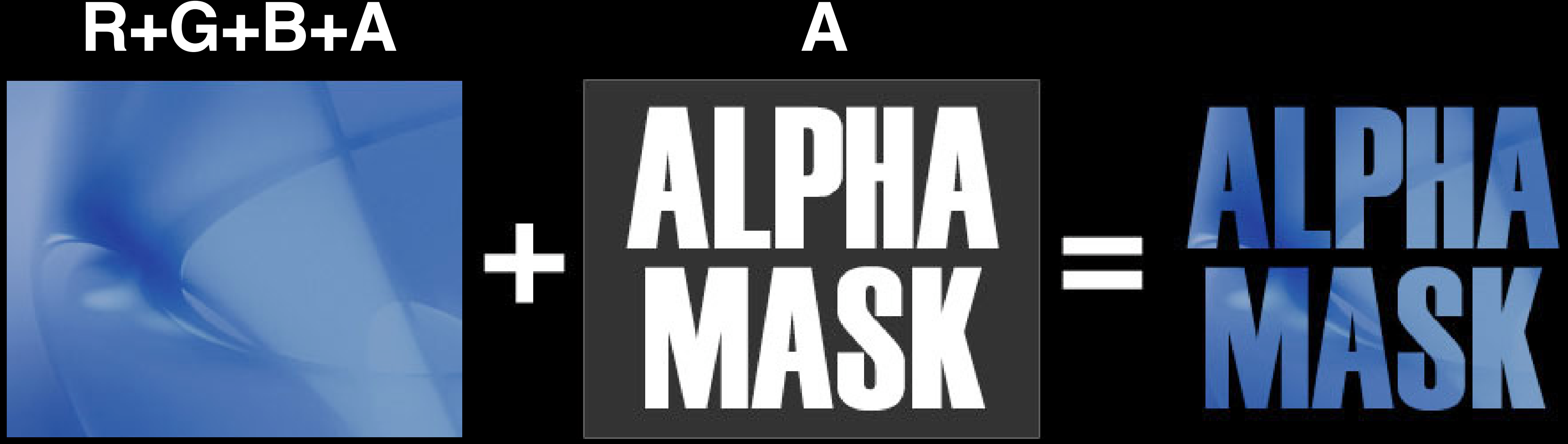
Standard OpenGL Blending

- OpenGL blend equation srcColor*srcFunc+ dstColor*dstFunc
- Typical examples
    - glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    - glBlendFunc(GL_ONE, GL_ONE_MINUS_SRC_ALPHA)

Multi-Texture Blending

- Potential for fill rate gain if you can use it
- May be easier to combine with other effects
- Internal format for masks important
    - **GL_ALPHA** for non-premultiplied alpha
    - **GL_INTENSITY** for premultiplied alpha

Multi Texture Blending



R+G+B+A

A

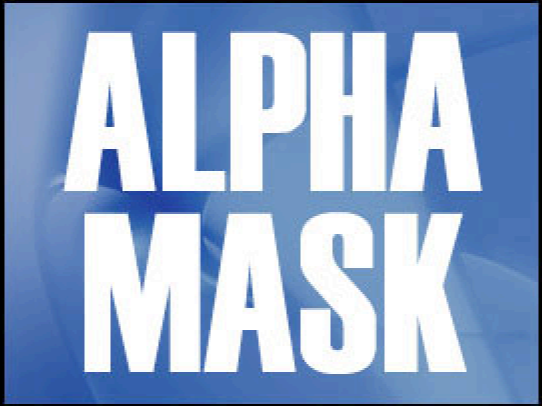
Destination Alpha Blending

- Use destination alpha as temporary mask storage
- Requires two passes for each layer with alpha
- Allows custom geometry to drive compositing

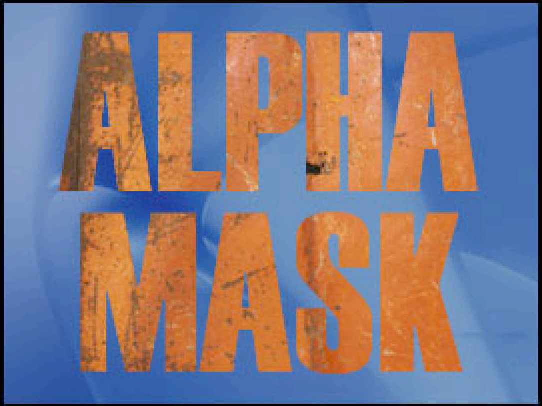
Destination Alpha Blending

1

2



3



Demo

OpenGL Blending

Image Effects

- Simple color scaling
    - Use GL primary color
    - **glColor4f(r, g, b, a)** with **GL_MODULATE** texture environment mode
        - Applies color and fade for whole layer

Image Warping

- Perspective warps

  http://www.cs.cmu.edu/~ph/869/www/notes/proj/proj.pdf
- OpenGL handles homogeneous coordinates

Color Correction

- Dependent 3D texture reads
    - R,G,B used as texture s,t,r coordinates
    - Computes (R’,G’,B’,A’) = F(R,G,B)
    - Requires **NV_texture_shader_3** or equivalent
- Background removal
- ColorSync correction

Demo

OpenGL Image Effects

OpenGL Compositor Lab

- Cocoa based
- Simple class hierarchy
    - Controller
    - CompositeGLView
    - Layer
        - MovieLayer
        - DVLayer
        - Quartz2DLayer
        - ImageLayer

OpenGL Compositor Lab

- Updates driven by layers
- Movies “just play”
    - Do not use SetMovieTime
- Compositor view redrawn lazily

Summary

- OpenGL is great for video and graphics integration
- OpenGL provides fast and flexible compositing and image manipulation
- Future hardware will let us do even more

Roadmap

500 Graphics and Imaging Overview	Room A2 **Tue., 10:30am**
503 Exploring the Quartz Compositor	Hall 2 **Tue., 3:30pm**
504 OpenGL: Graphics Programmability	Room A2 **Tue., 5:00pm**
505 OpenGL: Integrated Graphics I	Room J **Wed., 9:00am**

Roadmap

506 OpenGL: Integrated Graphics II	Room J **Wed., 10:30am**
509 ColorSync and Digital Media	Room C **Wed., 5:00pm**
511 Games Solutions: Graphics, Events, and Tidbits	Room C **Thurs., 10:30am**
512 Games Solutions: NetSprocket and OpenPlay	Room C **Thurs., 2:00pm**

Roadmap

513 OpenGL: Advanced 3D	Room J **Thurs., 3:30pm**
514 OpenGL: Performance and Optimization	Room J **Thurs., 5:00pm**
516 Graphics and Imaging Performance Tuning	Hall 2 **Fri., 3:30pm**
FF018 Graphics and Imaging:	Room J1 **Fri., 5:00pm**

Who to Contact

Sergio Mello
3D Graphics Technology Manager
sergio@apple.com

Q&A

Sergio Mello
3D Graphics Technology Manager
Worldwide Developer Relations
sergio@apple.com

http://developer.apple.com/wwdc2002/urls.html

WWDC2002

WWDC2002

WWDC2002